cosylab
CONTROL SYSTEM LABORATORY

Jožef Stefan Institute

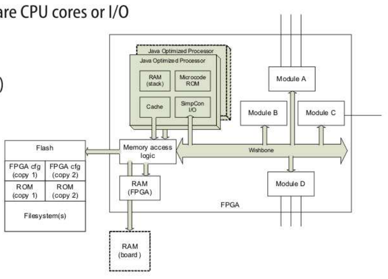
Beyond PCs: Accelerator Controls on Programmable Logic

Mark Pleško, Klemen Žagar, Jože Dedič

The large number of gates in modern FPGAs including processor cores allows implementation of complex designs, including a core implementing Java byte-code as the instruction set. Instruments based on FPGA technology are composed only of digital parts and are totally configurable. Based on experience gained on our products (a delay generators producing sub-nanosecond signals and function generators producing arbitrary functions of length in the order of minutes) and on our research projects (a prototype hardware platform for realtime Java, where Java runtime is the operating system and there is no need for Linux), I will speculate about possible future scenarios: A combination of an FPGA processor core and custom logic will provide all control tasks, slow and hard real-time, while keeping our convenient development environment for software such as Eclipse. I will illustrate my claims with designs for tasks such as low-latency PID controllers running at several dozen MHz, sub-nanosecond resolution timing, motion control and a versatile I/O controller - all implemented in real-time Java and on exactly the same hardware - just with different connectors.

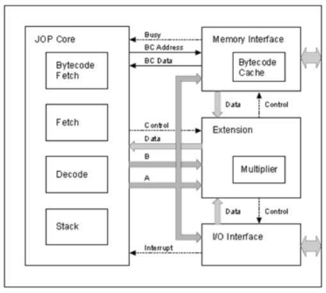
Hardware Arhitecture

- Architecture should be modular,where modules are CPU cores or I/O modules.
- Three-stage memory:
  - Fast, on-chip FPGA RAM (limited capacity)
  - Off-chip RAM (optional, large capacity)
  - Off-chip Flash (persistent storage)
- Flash contains:
  - FPGA configuration
  - Operating system and control software
  - File system
- Flash contains two copies of software (for fallback during software upgrade)
- Open on-chip bus (wishbone)
  - Many modules implemented in open source (Ethernet controllers, UART serial controllers, ...)

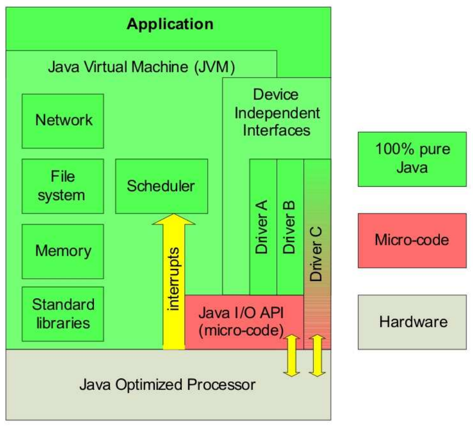


The CPU Core

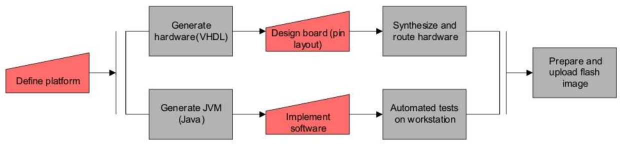
- Why not use Java as processor's native bytecode?
  - Advantage: high software development productivity with low defect rates due to good tool-chains (e.g., Eclipse IDE, static code checkers, well-established development methodologies, ...)
  - Disadvantage:
    - bytecode does not map ideally to hardware (stack vs. registers, ...)
    - real-time Java development is not yet mainstream
- Open cores implementing Java bytecode exist, e.g., Java Optimized Processor (JOP; http://jopdesign.com)

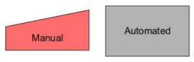


Software Architecture



Hardware-software co-design

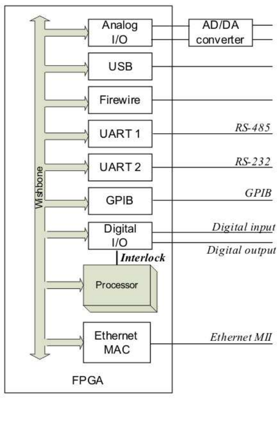

Define platform
Generate hardware(VHDL)
Design board (pin layout)
Synthesize and route hardware
Generate JVM (Java)
Implement software
Automated tests on workstation
Prepare and upload flash image



- Stages of development process
- Define platform: choose modules, CPU cores, ...
- Software:
  - Generate software: operating system core, drivers for modules, matching configuration of bus addresses, simulation environment
  - Implement software: full cycle of software implementation (design, implement, test)
  - Automated tests on workstation: code is run in simulation environment on developer's machine
- Hardware:
  - Generate hardware: VHDL files of modules are composed together, with bus addresses of modules properly configured
  - Design board: design board on which FPGA will be placed, map functionality to pins
  - Synthesize and route hardware: use of standard FPGA tools to produce bit image for FPGA configuration, and use of standard tools for board routing
- Note: software and hardware paths are independent and run in parallel. Interaction is needed only if platform is re-defined.

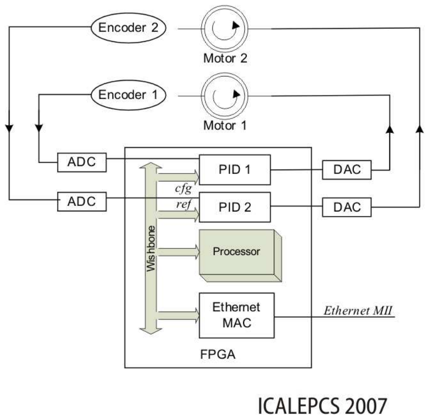
Example Application: Versatile I/O Controller

- High density, high-performance I/Os
  - Serial, GPIB, USB, firewire, …
  - Same board in different configurations
  - 10-100 I/Os per FPGA
- Software:
  - Control and configuration
  - Communication (with I/Os and/or SCADA)
- Hardware:
  - UARTs, USB controllers, …
  - Multiple cores
  - Dedicated DSPs (e.g., fireware camera image processing)
  - Interlocks on digital inputs



Example application: Motion Control

- Control of servo, stepper and other kinds of actuators
- Typically implemented with DSPs
  - Difficult to program and configure
- Co-design:
  - SW: motion programs, control, communication
  - HW: PID loops (up to several MHz)


Encoder 2
Motor 2
Encoder 1
Motor 1
ADC
ADC
PID 1
PID 2
cfg
ref
DAC
DAC
Wishbone
Processor
Ethernet MAC
Ethernet MII
FPGA